REMARKS

Claims 1-24 are currently pending in the subject application and are presently under consideration. Claims 1, 2, 6, 9-12 and 21-24 have been amended herein. A complete listing of the claims showing changes made can be found at 4-7. Amendments to the specification are located at pages 2 and 3. Favorable reconsideration of the subject patent application is respectfully requested in view of the comments and amendments herein.

I. Objection to Claims 24 for an Informality

Claim 24 stands objected to for a minor informality. Claim 24 has been amended to cure the subject informality, namely "carry out" has been amended to recite "carrying out." Accordingly, this objection should be withdrawn.

II. Objection to Claims 1 and 12 Under 35 U.S.C. § 112, First Paragraph

Claims 1, 9-12, and 21-24 stand objected to under 35 U.S.C. § 112, first paragraph as failing to comply with the enablement requirement. The manner and process of making the claimed feature is alleged to be unclear. In particular, the Examiner asserts that it is not clear what application preference classes are and how they are to be extended. Withdrawal of this objection is respectfully requested for at least the following reasons.

Application preferences classes are clearly described in the subject application. Preference classes are units of information agent schema development that can include, *inter alia*, a set of condition and action classes. (*See* page 26, line 3 and the Appendix). The information agent schema or logic schema defines logic building blocks or templates that can be put together by a user. In effect, the logic schema constrains the actual expressive power of end-user logic, thereby making it practical for untrained end-users to actually "program" an application. (page 18, lines 19-23). Thus, end-users can specify preferences that are based on the structure provided by the preference classes. As described, preferences are units of end-user logic in the form of "ON (event) IF (condition) THEN (action set)." Preferences can have several properties. For example, the preference should belong to (or instantiate) a preference class. Further, the condition

should be a declarative Boolean expression combining one or more instances of condition classes, wherein every condition instance defines parameter values for a condition class. Every action set should be a set of action classes, every action instance defining parameter values for an action class. (*See*, page 26, line 24 through page 27, line 2).

As disclosed by the specification, it is desirable to allow information agent applications to be extensible. Extensibility is primarily aimed at enabling new conditions and actions to be added to an application after the application is installed. The degree to which an IA application is extensible can be determined by the extent to which new conditions and actions are made available to users defining new preferences within the context of an existing application. To understand how this is done, it is important to emphasize the evolutionary chain by which definition of an action or condition function eventually becomes accessible to end-users of an application. Conditions and actions begin as condition or action functions. Between the time when new condition or action function is defined and when the function is bound into an existing application by declaration of a corresponding condition or action, the function can be considered a candidate function. Conditions or actions are candidates for use by existing applications to be extended such that the application can use the conditions and/or actions. Once an application binds its preference classes to a condition or logic, function candidate conditions or actions simply become conditions or actions. Finally, when an end-user utilizes a condition or action within the context of a defined preference, that action or condition is said to be instantiated. (*See*, page 98, line 21 through page 97, line 12). Preference classes can be extended by binding new condition or action functions provided by other applications to preference class declarations.

Claim 1 recites a first executable application including one or more functions that are registered in a registry component and an extension component that reads function data from the registry and binds a second executable application to the first, wherein second application preference class declarations are bound to the functions provided by the first application.

Claim 12 recites a method comprising receiving an extension data file (EDF) containing information about candidate function bindings, registering one or more function bindings in a central data store, and binding a function of a first executable

application to a second executable application utilizing binding function information located in central data store.

In view of the aforementioned, it is readily apparent that the preferences classes and the extension thereof are enabled by the subject disclosure. Accordingly, this objection should be withdrawn.

III. Objection to Claims 9-11 Under 35 U.S.C. § 112, First Paragraph

Claims 9-11 stand objected to under 35 U.S.C. § 112, first paragraph as failing to comply with the enablement requirement. Specifically, the Examiner asserts that the meaning of "extending condition constants," "first order constant," and "Nth order constant" is unclear. Withdrawal of this objection is respectfully requested for at least the following reasons.

For purposes of clarity, claims 9-11 have been amended to recite "constant accessors" rather than simply "constants." Constant accessors are named groups of objects that provide arguments to conditions and actions in place of a user having to manually specify each such object. These constants are simply names veneered over functions that operate to find and materialize the correct information, namely the members of the group associated with the name of the constant. (*See*, page 29, lines 18-25). Claim 10 recites a first order constant accessor, and claim 11 recites an Nth order constant accessor. A first order constant accessor refers to a single accessor, for example "MyFamily." An Nth order constant accessor is as a combination of constant named groups constructed similar in semantics to prepositional phrases, for example "FriendsOfMyFamily" or "EmailsFromPreferredCustomersInAppointmentsToday." (*See* page 31, lines 3-19).

Hence, claims 9-11 comply with the enablement requirement. Accordingly, this objection should be withdrawn.

IV. Objection to Claims 21-24 Under 35 U.S.C. § 112, First Paragraph

Claims 21-24 stand objected to under 35 U.S.C. § 112, first paragraph as failing to comply with the enablement requirement. Specifically, the Examiner asserts that the meaning of "extending condition constants," "first order constant," and "Nth order

constant" is unclear. Withdrawal of this objection is respectfully requested for at least the following reasons.

Claims 21-23 have been amended to recite "constant accessors" rather then simply "constants" for purposes of clarity. Constant accessors are named groups of objects that provide arguments to conditions and actions in place of a user having to manually specify each such object. These constants are simply names veneered over functions that operate to find and materialize the correct information, namely the members of the group associated with the name of the constant. (*See*, page 29, lines 18-25). Claim 22 recites a first order constant accessor, and claim 23 recites an Nth order constant accessor. A first order constant accessor refers to a single accessor, for example "MyFamily." An Nth order constant accessor is as a combination of constant named groups constructed similar in semantics to prepositional phrases, for example "FriendsOfMyFamily" or "EmailsFromPreferredCustomersInAppointmentsToday." (*See* page 31, lines 3-19).

Claim 24 depends from claim 21 and thus includes all the features of claim 24. Accordingly, the objection to claims 21-24 should be withdrawn.

V. Objection to Claim 1 Under 35 U.S.C. § 112, Second Paragraph

Claim 1 stands objected to under 35 U.S.C. § 112, second paragraph as failing to point out and distinctly claim the subject matter that applicant regards as his invention. Claim 1 has been amended to cure the deficiency. In particular, "register component" has been amended to recite "registry component" for purposes of element consistency. According, this objection should be withdrawn.

VI. Rejection of Claims 1-2, 6-8, and 12-14 Under 35 U.S.C. § 103(a)

Claims 1-2, 2-8, and 12-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over "IBM Ada/6000," September 2, 1998, in view of Shaughnessy, *et al.* (U.S. 6,026,235) and Walker (U.S. 6,016,394). This rejection should be withdrawn for at least the following reasons.

The cited references alone or in combination fail to teach or suggest each and every feature recited by the claims. Moreover, at least "IBM Ada/6000" and Shaughnessy, *et al.*

teach away from the claimed invention.

> A prima facie case of obviousness is established by a showing of three basic criteria. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the ***prior art reference*** (or references when combined) ***must teach or suggest all the claim limitations.*** *See* MPEP §706.02(j). The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *See In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991) (emphasis added).

> ***A reference that teaches away from the art is a per se demonstration of a lack of prima facie obviousness.*** *In re Geisler*, 116 F.3d 1465, 43 USPQ2d 1362 (Fed. Cir. 1997) (emphasis added). A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant. *In re Gurley*, 27 F.3d 551, 553, 31 USPQ2d 1130 (Fed. Cir. 1994); *Tec Air, Inc. v. Denso Mfg. Mich. Inc.*, 192 F.3d 1353, 52 USPQ2d 1294 (Fed. Cir. 1999).

Independent claim 1, recites an extension component that ... binds a second ***executable application to the first***, wherein second application ***preference class declarations are bound to the functions provided by the first application.*** Independent claim 12, recites binding a function of a first ***executable application*** to a ***preference class of a second executable application*** utilizing binding function information located in the central data store. IBM Ada/6000, Shaughnessy, *et al.* and Walker alone or in combination fail to teach or suggest these claim features. In fact, both "IBM Ada/6000" and Shaughnessy, *et al.* pertain to compilation and linking. The compilation and linking processes includes combining and linking one or more modules to ***produce a machine executable program***. The subject invention, as claimed, utilizes executable programs or applications and functions provided thereby to enhance other application capabilities, specifically by extending preference classes to allow use of such functions. By way of example, assume application A has a context

condition DoNotDisturb() that can be used when defining preferences regarding when and how a user is to be interrupted. Assume that there is an application B that installs a new action SendEmail(). The application B can make the SendEmail() action or function available to application A by providing a binding registered in a registry that is accessible to both application A and B. The binding can then be read from the registry and applied to application A such that users defining new preferences in application A will be able to use the new action SendEmail(). As a result, a user may create a preference that sends an email to him/her as a result of an attempt to contact the user. As provided in the specification, "extensibility is primarily aimed at enabling new conditions and actions to be added to an application subsequent to the time at which the application is installed, without further intervention by the author(s) of the original application." (page 96, lines 19-21). ***It is not a goal of extensibility to create a whole new executable application. Rather, it is to enable dynamic extensions to the layer of the system that allows users to specify preference logic.*** (*See* page 95, lines 1-9). Thus, not only do "IBM Ada/6000" and Shaughnessy, *et al.* fail to teach or suggest the claimed features of the invention, they actually teach away from them by teaching generation of a new executable.

Walker does not make up for the deficiencies of "IBM Ada/6000" and Shaughnessy, *et al.* Furthermore, Walker does not teach or suggest application preferences as alleged by the Examiner. In particular, the Examiner relies in part on smart codes and the disclosure that "By using smart codes default values may be specified as preferences for use in application creation." The subject invention as claimed is not concerned with application creation *per se*. Rather, it pertains to enhancing or extending existing applications by binding functions (*e.g.*, actions, conditions...) from one application to the preference class of at least one other application. In this manner, the preferences that can be specified by a user can be extended by making more functions available thereto. Thus, Walker does not teach or suggest preferences or preference classes as claimed.

In light of the aforementioned, it is readily apparent that claims 1 and claims 12 are not made obvious by "IBM Ada/6000," Shaughnessy, *et al.* and Walker alone or in combination. Accordingly, the rejection of claims 1 and 12 (as well as claims 2, 6-8, and 13-14 depending therefrom) should be withdrawn.

VII. Rejection of Claim 3-5 Under 35 U.S.C. §103(a)

Claims 3-5 stand rejected under 35 U.S.C. §103(a) as being unpatentable over "IBM Ada/6000," September 2, 1998 and Shaughnessy, *et al.* (U.S. 6,026,235) in view of Walker (U.S. 6,016,394) further in view of "Compile Time Scheduling of an Ada Subset" by E.W. Gierring III, *et al.* This rejection should be withdrawn for at least the following reason.

Claims 3-5 depend from claim 1 and therefore include all the features of claim 1. Claim 1 is allowable for at this the reasons provided *supra*. The article "Compile Time Scheduling of an Ada Subset" does not make up for the deficiencies of "IBM Ada/6000," Shaughnessy, *et al.* and Walker. Accordingly, claims 2-5 are allowable for at least the same reasons as claim 1.

Furthermore, it should be appreciated that the features recited by claims 3-5 provide separate bases for patentability over and above those presented with respect to independent claim 1 from which they depend. For instance, the examiner alleges that accessors, as recited by claim 5, are taught by the article "Compile Time Scheduling of an Ada Subset" from the disclosure that any task can call an entry. As provided in the specification accessors or constant accessors are named groups of objects that provide arguments to conditions and actions in place of a user having to manually specify each such object. These constants are simply names veneered over functions that operate to find and materialize the correct information, namely the members of the group associated with the name of the constant. (*See*, page 29, lines 18-25). It is readily apparent that accessors are not taught or suggested by any of the cited references alone or in combination.

For at least the above reasons, the rejection of claims 3-5 should be withdrawn.

VIII. Rejection of Claims 15-20 Under 35 U.S.C. §103(a)

Claims 15-20 stand rejected under 35 U.S.C. §103(a) as being unpatentable over "IBM Ada/6000," September 2, 1998 and Shaughnessy, *et al.* (U.S. 6,026,235). This rejection should be withdrawn for at least the following reasons.

As per claim 15, neither "IBM Ada/6000" nor Shaughnessy, *et al.* disclose, teach or suggest notifying dependent applications, as recited by claim 15. The Examiner asserts that this feature is taught by "IBM Ada/6000" because the article teaches displaying an obscure error message. Applicants' representative respectfully disagrees. The mere generation of an

error message does not teach notifying dependent applications. In fact, "IBM Ada/6000" does not even pertain to uninstalling an application. "IBM Ada/6000" teaches displaying an error message to a user when a compiler aborts. There is absolutely no teaching or suggestion of notifying dependent applications, as claimed. Accordingly, the rejection of claim 15 (as well as claims 16-20 depending therefrom) should be withdrawn.

CONCLUSION

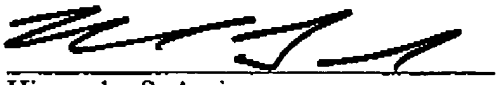
The present application is believed to be in condition for allowance in view of the above comments and amendments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP545USA].

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact applicants' undersigned representative at the telephone number below.

Respectfully submitted,

AMIN & TUROCY, LLP



Himanshu S. Amin
Reg. No. 40,894

AMIN & TUROCY, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731